Derivatives for Regular Shuffle Expressions

Martin Sulzmann[1] and Peter Thiemann[2]

[1] Faculty of Computer Science and Business Information Systems
Karlsruhe University of Applied Sciences
Moltkestrasse 30, 76133 Karlsruhe, Germany
martin.sulzmann@hs-karlsruhe.de
[2] Faculty of Engineering, University of Freiburg
Georges-Köhler-Allee 079, 79110 Freiburg, Germany
thiemann@acm.org

Abstract. There is a rich variety of shuffling operations ranging from asynchronous interleaving to various forms of synchronizations. We introduce a general shuffling operation which subsumes earlier forms of shuffling. We further extend the notion of a Brzozowski derivative to the general shuffling operation and thus to many earlier forms of shuffling. This extension enables the direct construction of automata from regular expressions involving shuffles that appear in specifications of concurrent systems.

Keywords: automata and logic, shuffle expressions, derivatives

1 Introduction

We consider an extension of regular expressions with a binary shuffle operation which bears similarity to shuffling two decks of cards. Like the extension with negation and complement, the language described by an expression extended with shuffling remains regular. That is, any expression making use of shuffling can be expressed in terms of basic operations such as choice, concatenation and Kleene star. However, the use of shuffling yields a much more succinct representation of problems that occur in modeling of concurrent systems [4, 12].

Our interest is to extend the notion of a Brzozowski derivative [3] to regular expressions with shuffles. Derivatives support the elegant and efficient construction of automata-based word recognition algorithms [10] and are also useful in the development of related algorithms for equality and containment among regular expressions [1, 6].

Prior work in the area To the best of our knowledge, there is almost no prior work which studies the notion of Brzozowski derivatives in connection with shuffling. We are only aware of one work [9] which appears to imply a definition of derivatives for strongly synchronized shuffling [2]. Further work in the area studies the construction of automata for a specific form of shuffling commonly referred to as asynchronous interleaving [5, 7]. In contrast, we provide detailed

definitions how to obtain derivatives including formal results for various shuffling operations [11, 4, 2]. Our results imply algorithms for constructing automata as well as for checking equality and containment of regular expressions with shuffles.

Contributions After introducing our notation in Section 2 and reviewing existing variants of shuffling in Section 3, we claim the following contributions:

- We introduce a general shuffle operation which subsumes previous forms of shuffling (Section 4).
- We extend the notion of Brzozowski derivatives to the general shuffle operation and are able to re-establish all of its "good" properties (Section 5).
- Based on the general shuffle operation, we provide systematic methods to obtain derivatives for specific variants of shuffling (Section 5.1).

We conclude in Section 6.

2 Preliminaries

Let Σ be a fixed alphabet (i.e., a finite set of symbols). We usually denote symbols by x, y and z. The set Σ^* denotes the set of finite words over Σ. We write Γ, Δ to denote subsets (sub-alphabets) of Σ. We write ϵ to denote the empty word and $v \cdot w$ to denote the concatenation of two words v and w. We generally write $L_1, L_2 \subseteq \Sigma^*$ to denote languages over Σ.

We write $L_2 \backslash L_1$ to denote the left quotient of L_1 with L_2 where $L_2 \backslash L_1 = \{w \mid \exists v \in L_2 . v \cdot w \in L_1\}$. We write $x \backslash L$ as a shorthand for $\{x\} \backslash L$.

We write $\alpha(w)$ to denote the set of symbols which appear in a word. The inductive definition is as follows: (1) $\alpha(\epsilon) = \emptyset$, (2) $\alpha(x \cdot w) = \alpha(w) \cup \{x\}$. The extension to languages is as follows: $\alpha(L) = \bigcup_{w \in L} \alpha(w)$.

We write $\Pi_\Gamma(w)$ to denote the projection of a word w onto a sub-alphabet Γ. The inductive definition is as follows:

$$\Pi_\Gamma(\epsilon) = \epsilon \qquad \Pi_\Gamma(x \cdot w) = \begin{cases} x \cdot \Pi_\Gamma(w) & x \in \Gamma \\ \Pi_\Gamma(w) & x \notin \Gamma \end{cases}$$

3 Shuffling Operations

Definition 1 (Shuffling). *The* shuffle operator $\| :: \Sigma^* \times \Sigma^* \to \wp(\Sigma^*)$ *is defined inductively as follows:*

$$\begin{aligned} \epsilon \| w &= \{w\} \\ w \| \epsilon &= \{w\} \\ x \cdot v \| y \cdot w &= \{x \cdot u \mid u \in v \| y \cdot w\} \cup \{y \cdot u \mid u \in x \cdot v \| w\} \end{aligned}$$

We lift shuffling to languages by $L_1 \| L_2 = \{u \mid u \in v \| w \wedge v \in L_1 \wedge w \in L_2\}$.

For example, we find that $x \cdot y \| z = \{x \cdot y \cdot z, x \cdot z \cdot y, z \cdot x \cdot y\}$.

While the shuffle operator represents the asynchronous interleaving of two words $v, w \in \Sigma^*$, there are also shuffle operators that include some synchronization. The strongly synchronized shuffle of two words w.r.t. some sub-alphabet Γ imposes the restriction that the traces must synchronize on all symbols in Γ. All symbols not appearing in Γ are shuffled. In its definition, we write $(p) \Rightarrow X$ for: if p then X else $\emptyset$.

Definition 2 (Strong Synchronized Shuffling). *The* synchronized shuffling operator *w.r.t.* $\Gamma \subseteq \Sigma$, $|||_\Gamma :: \Sigma^* \times \Sigma^* \to \wp(\Sigma^*)$*, is defined inductively as follows.*

$$
\begin{array}{lll}
\epsilon|||_\Gamma w & = (\Gamma \cap \alpha(w) = \emptyset) \Rightarrow \{w\} & \textit{(S1)} \\
w|||_\Gamma \epsilon & = (\Gamma \cap \alpha(w) = \emptyset) \Rightarrow \{w\} & \textit{(S2)} \\
x \cdot v|||_\Gamma y \cdot w & = (x = y \wedge x \in \Gamma) \Rightarrow \{x \cdot u \mid u \in v|||_\Gamma w\} & \textit{(S3)} \\
& \cup (x \notin \Gamma) \Rightarrow \{x \cdot u \mid u \in v|||_\Gamma y \cdot w\} & \textit{(S4)} \\
& \cup (y \notin \Gamma) \Rightarrow \{y \cdot u \mid u \in x \cdot v|||_\Gamma w\} & \textit{(S5)}
\end{array}
$$

We lift strongly synchronized shuffling to languages by

$$L_1|||_\Gamma L_2 = \{u \mid u \in v|||_\Gamma w \wedge v \in L_1 \wedge w \in L_2\}$$

The base cases (S1) and (S2) impose the condition (via $\Gamma \cap \alpha(w) = \emptyset$) that none of the symbols in w shall be synchronized. If the condition is violated we obtain the empty set. For example, $\epsilon|||_{\{x\}} y \cdot z = \{y \cdot z\}$, but $\epsilon|||_{\{x\}} x \cdot y \cdot z = \emptyset$.

In the inductive step, a symbol in Γ appearing on both sides forces synchronization (S3). If the leading symbol on either side does not appear in Γ, then it can be shuffled arbitrarily. See cases (S4) and (S5). These three cases ensure progress until one side is reduced to the empty string. For example, we find that $x \cdot y|||_{\{x\}} x \cdot z = \{x \cdot y \cdot z, x \cdot z \cdot y\}$. On the other hand, $x \cdot y|||_{\{x,y\}} x \cdot z = \emptyset$.

Shuffling and strongly synchronized shuffling correspond to the *arbitrary* synchronized shuffling and strongly synchronized shuffling operations by Beek and coworkers [2]. The inductive definitions that we present simplify our proofs.

Beek and coworkers [2] also introduce a *weak* synchronized shuffling operation. In its definition, we write $L \cdot x$ as a shorthand for $\{w \cdot x \mid w \in L\}$ and $x \cdot L$ as a shorthand for $\{x \cdot w \mid w \in L\}$

Definition 3 (Weak Synchronized Shuffling). *Let* $v, w \subseteq \Sigma^*$ *and* $\Gamma \subseteq \Sigma$*. Then, we define*

$$
\begin{array}{l}
v| \sim |_\Gamma w = \{u \mid \exists n \geq 0, x_i \in \Gamma, v_i \in \Gamma, w_i \in \Gamma. \\
\qquad v = v_1 \cdot x_1 ... x_n \cdot v_{n+1} \wedge w = w_1 \cdot x_1 ... x_n \cdot w_{n+1} \wedge \\
\qquad u \in (v_1 \| w_1) \cdot x_1 ... x_n \cdot (v_{n+1} \| w_{n+1}) \\
\qquad \alpha(v_i) \cap \alpha(w_i) \cap \Gamma = \emptyset\}
\end{array}
$$

and for languages: $L_1| \sim |_\Gamma L_2 = \{u \mid u \in v| \sim |_\Gamma w \wedge v \in L_1 \wedge w \in L_2\}$.

The weak synchronized shuffle of two words v and w synchronizes only on those symbols in Γ that occur in both v and w. For example, $x \cdot y| \sim |_{\{x,y\}} x \cdot z = \{x \cdot y \cdot z, x \cdot z \cdot y\}$ because $y \notin \alpha(x \cdot z)$ whereas $x \cdot y|||_{\{x,y\}} x \cdot z = \emptyset$.

There is another variant of synchronous shuffling called *synchronous composition* [11, 4]. The difference to strongly synchronized shuffling is that synchronization occurs on symbols common to both operands. Thus, synchronous composition can be defined by projecting onto the symbols of the operands.

Definition 4 (Synchronous Composition). *The synchronous composition operator $|||$ is defined by:*

$$L_1|||L_2 = \{w \in (\alpha(L_1) \cup \alpha(L_2))^* \mid \Pi_{\alpha(L_1)}(w) \in L_1 \wedge \Pi_{\alpha(L_2)}(w) \in L_2\}$$

For example, $x \cdot y|||x \cdot z$ equals $\{x \cdot y \cdot z, x \cdot z \cdot y\}$.

It turns out that the strong synchronized shuffling operation $|||_\Gamma$ subsumes synchronous composition due to the customizable set Γ. In Section 4, we show an even stronger result: All of the shuffling variants we have seen can be expressed in terms of a general synchronous shuffling operation.

4 General Synchronous Shuffling

The general synchronous shuffling operation is parameterized by a set of synchronizing symbols, Γ, and two additional sets P_1 and P_2 that keep track of 'out of sync' symbols from Γ. We write $X \cup x$ as a shorthand for $X \cup \{x\}$.

Definition 5 (General Synchronous Shuffling). *Let $\Gamma, P_1, P_2 \subseteq \Sigma$. The general synchronous shuffling operator ${}^{P_1}||^{P_2}_\Gamma :: \Sigma^* \times \Sigma^* \to \wp(\Sigma^*)$ is defined inductively as follows.*

$$\begin{aligned}
\epsilon {}^{P_1}||^{P_2}_\Gamma w &= ((\alpha(w) \cap \Gamma = \emptyset) \vee (P_1 \cap (P_2 \cup \alpha(w)) = \emptyset)) \Rightarrow \{w\} && \text{(G1)}\\
w {}^{P_1}||^{P_2}_\Gamma \epsilon &= ((\alpha(w) \cap \Gamma = \emptyset) \vee (P_1 \cup \alpha(w)) \cap P_2 = \emptyset)) \Rightarrow \{w\} && \text{(G2)}\\
x \cdot v {}^{P_1}||^{P_2}_\Gamma y \cdot w &= (x \notin \Gamma) \Rightarrow \{x \cdot u \mid u \in v {}^{P_1}||^{P_2}_\Gamma y \cdot w\} && \text{(G3)}\\
&\quad \cup (y \notin \Gamma) \Rightarrow \{y \cdot u \mid u \in x \cdot v {}^{P_1}||^{P_2}_\Gamma w\} && \text{(G4)}\\
&\quad \cup (x = y \wedge x \in \Gamma \wedge P_1 \cap P_2 = \emptyset) \Rightarrow \{x \cdot u \mid u \in v {}^{\emptyset}||^{\emptyset}_\Gamma w\} && \text{(G5)}\\
&\quad \cup (x = y \wedge x \in \Gamma \wedge P_1 \cap P_2 \neq \emptyset) \Rightarrow \{x \cdot u \mid u \in v {}^{P_1}||^{P_2}_\Gamma w\} && \text{(G6)}\\
&\quad \cup (x \in \Gamma \wedge (P_1 \cup x) \cap P_2 = \emptyset) \Rightarrow \{x \cdot u \mid u \in v {}^{P_1 \cup x}||^{P_2}_\Gamma y \cdot w\} && \text{(G7)}\\
&\quad \cup (y \in \Gamma \wedge P_1 \cap (P_2 \cup y) = \emptyset) \Rightarrow \{y \cdot u \mid u \in x \cdot v {}^{P_1}||^{P_2 \cup y}_\Gamma w\} && \text{(G8)}
\end{aligned}$$

For $L_1, L_2 \subseteq \Sigma^*$, $\Gamma \subseteq \Sigma$, $P_1, P_2 \subseteq \Sigma$ we define

$$L_1 {}^{P_1}||^{P_2}_\Gamma L_2 = \{u \mid u \in v {}^{P_1}||^{P_2}_\Gamma w \wedge v \in L_1 \wedge w \in L_2\}$$

The definition of general synchronous shuffling is significantly more involved compared to the earlier definitions. The various cases are necessary to encode the earlier shuffle operations from Section 3. The exact purpose of the individual cases will become clear shortly.

In our first result we observe that ${}^{\Sigma}||^{\Sigma}_\Gamma$ exactly corresponds to strongly synchronized shuffling ($|||_\Gamma$).

Theorem 1. *For any $L_1, L_2 \subseteq \Sigma^*$ and $\Gamma \subseteq \Sigma$: $L_1|||_\Gamma L_2 = L_1 {}^{\Sigma}||^{\Sigma}_\Gamma L_2$.*

Proof. We choose a 'maximal' assignment for P_1 and P_2 such that definition of ${}^{P_1}||^{P_2}_{\Gamma}$ reduces to $|||_{\Gamma}$. We observe that property $P_1 = \Sigma \wedge P_2 = \Sigma$ (SP) is an invariant. For cases (G3-4) and (G6) the invariant property clearly holds. For cases (G5), (G7-8) the preconditions are violated. Hence, for ${}^{\Sigma}||^{\Sigma}_{\Gamma}$ only cases (G1-4) and (G6) will ever apply.

Under the given assumptions, we can relate the cases in Definition 2 and Definition 5 as follows. Cases (G1-2) correspond to cases (S1-2). Cases (G3-4) correspond to cases (S4-5). Case (G6) corresponds to case (S3). Due to the invariant property (SP) cases (G5) and (G7-8) never apply.

Hence, ${}^{\Sigma}||^{\Sigma}_{\Gamma}$ and $|||_{\Gamma}$ yield the same result. □

Via similar reasoning we can show that for $\Gamma = \emptyset \wedge P_1 = \emptyset \wedge P_2 = \emptyset$ general synchronized shuffling boils down to (arbitrary) shuffling.

Theorem 2. *For any* $L_1, L_2 \subseteq \Sigma^*$*:* $L_1 \| L_2 = L_1{}^{\emptyset}||^{\emptyset}_{\emptyset} L_2$.

An immediate consequence from Theorem 1 (set Σ and Γ to $\emptyset$) and Theorem 2 is that shuffling can also expressed in terms of strong synchronized shuffling.

Corollary 1. *For any* $L_1, L_2 \subseteq \Sigma^*$*:* $L_1 \| L_2 = L_1 |||_{\emptyset} L_2$.

Our next result establishes a connection to weak synchronized shuffling [2].

Theorem 3. *For any* $L_1, L_2 \subseteq \Sigma^*$*,*$\Gamma \subseteq \Sigma$*:* $L_1| \sim |_{\Gamma} L_2 = L_1{}^{\emptyset}||^{\emptyset}_{\Gamma} L_2$.

Proof. Property $P_1, P_2 \subseteq \Gamma \wedge P_1 \cap P_2 = \emptyset$ (WP) is an invariant of ${}^{P_1}||^{P_2}_{\Gamma}$. For cases (G3-4) the invariant property clearly holds. More interesting are cases (G7-8) where P_1, resp., P_2 is extended. Under the precondition property (WP) remains invariant. Case (G6) never applies. Case (G5) clearly maintains the invariant.

Recall that for strong synchronized shuffling the roles of (G5) and (G6) are switched. See proof of Theorem 1. This shows that while cases (G5) and (G6) look rather similar both are indeed necessary.

As we can see, under the invariant condition (WP), the purpose of P_1, P_2 is to keep track of "out of sync" symbols from Γ. See cases (G7-8). Case (G5) synchronizes on $x \in \Gamma$. Hence, P_1 and P_2 are (re)set to $\emptyset$.

Thus, we can show (via some inductive argument) that ${}^{\emptyset}||^{\emptyset}_{\Gamma}$ under the (WP) invariant corresponds to weak synchronized shuffling as defined in Definition 3. □

It remains to show that synchronous composition is subsumed by general synchronized shuffling. First, we verify that synchronous composition is subsumed by strongly synchronous shuffling.

Theorem 4. *For any* $L_1, L_2 \subseteq \Sigma^*$ *we find that* $L_1 ||| L_2 = L_1 |||_{\alpha(L_1) \cap \alpha(L_2)} L_2$.

Proof. To establish the direction $L_1 ||| L_2 \supseteq L_1 |||_{\alpha(L_1) \cap \alpha(L_2)} L_2$ we verify that the projection of a strongly synchronizable word w.r.t. $\alpha(L_1) \cap \alpha(L_2)$ yields words in the respective languages L_1 and L_2.

Formally: Let $u, v, w \in \Sigma^*$, $L_1, L_2 \subseteq \Sigma^*$, $\Gamma \subseteq \Sigma$ such that $u \in v|||_\Gamma w$ where $v \in L_1$, $w \in L_2$ and $\Gamma \subseteq \alpha(L_1) \cap \alpha(L_2)$. Then, we find that (1) $\Pi_{\alpha(L_1)}(u) = v$ and (2) $\Pi_{\alpha(L_2)}(u) = w$. The proof of this statement proceeds by induction over u.

Direction $L_1|||L_2 \supseteq L_1|||_{\alpha(L_1)\cap\alpha(L_2)}L_2$ can be verified similarly. We show that if the projection of a word onto $\alpha(L_1)$ and $\alpha(L_2)$ yields words in the respective language, then the word must be strongly synchronizable w.r.t. $\alpha(L_1) \cap \alpha(L_2)$.

Formally: Let $u, v, w \in \Sigma^*$, $L_1, L_2 \subseteq \Sigma^*$ such that $w \in (\alpha(L_1) \cup \alpha(L_2))^*$, $\Pi_{\alpha(L_1)}(w) = u \in L_1$ and $\Pi_{\alpha(L_1)}(w) = v \in L_2$. Then, we find that $w \in u|||_{\alpha(L_1)\cap\alpha(L_2)}v$. The proof proceeds again by induction, this time over w. □

An immediate consequence of Theorem 1 and Theorem 4 is the following result. Synchronous composition is subsumed by general synchronous shuffling.

Corollary 2. *For any $L_1, L_2 \subseteq \Sigma^*$ we find that $L_1|||L_2 = {L_1}^{\Sigma}||^{\Sigma}_{\alpha(L_1)\cap\alpha(L_2)}L_2$.*

5 Derivatives for General Synchronous Shuffling

Brzozowski derivatives [3] are a useful tool to translate regular expressions into finite automata and to obtain decision procedures for equivalence and containment for regular expressions. We show that Brzozowski's results and their applications can be extended to regular expressions that contain shuffle operators. Based on the results of the previous section, we restrict our attention to regular expressions extended with the general synchronous shuffle operator.

Definition 6. *The set R_Σ of* regular shuffle expressions *is defined inductively by $\phi \in R_\Sigma$, $\epsilon \in R_\Sigma$, $\Sigma \subseteq R_\Sigma$, and for all $r, s \in R_\Sigma$ and $\Gamma, P_1, P_2 \subseteq \Sigma$ we have that $r + s$, $r \cdot s$, r^*, $r^{P_1}||^{P_2}_\Gamma s \in R_\Sigma$.*

Definition 7. *The language $\mathcal{L}() : R_\Sigma \to \Sigma^*$ denoted by a regular shuffle expression is defined inductively as follows. $\mathcal{L}(\phi) = \emptyset$. $\mathcal{L}(\epsilon) = \{\epsilon\}$. $\mathcal{L}(x) = \{x\}$. $\mathcal{L}(r+s) = \mathcal{L}(r) \cup \mathcal{L}(s)$. $\mathcal{L}(r \cdot s) = \{v \cdot w \mid v \in \mathcal{L}(r) \wedge w \in \mathcal{L}(s)\}$. $\mathcal{L}(r^*) = \{w_1 ... w_n \mid n \geq 0 \wedge w_i \in \mathcal{L}(r) \wedge i \in \{1, ..., n\}\}$. $\mathcal{L}(r^{P_1}||^{P_2}_\Gamma s) = \{u \mid u \in v^{P_1}||^{P_2}_\Gamma w \wedge v \in \mathcal{L}(r) \wedge w \in \mathcal{L}(s)\}$.*

An expression r is *nullable* if $\epsilon \in \mathcal{L}(r)$. The following function $n(\_)$ detects nullable regular expressions.

Definition 8. *We define $n(\_) : R_\Sigma \to Bool$ inductively as follows. $n(\phi) = false$. $n(\epsilon) = true$. $n(x) = false$. $n(r + s) = n(r) \vee n(s)$. $n(r \cdot s) = n(r) \wedge n(s)$. $n(r^*) = true$. $n(r^{P_1}||^{P_2}_\Gamma s) = n(r) \wedge n(s)$.*

Lemma 1. *For all $r \in R_\Sigma$ we have that $\epsilon \in \mathcal{L}(r)$ iff $n(r) = true$.*

Proof. The proof proceeds by induction over r. For brevity, we only consider the shuffle case as the remaining cases are standard. $\epsilon \in \mathcal{L}(r^{P_1}||^{P_2}_\Gamma s)$ iff (by definition) $\epsilon \in v^{P_1}||^{P_2}_\Gamma w$ for some $v \in \mathcal{L}(r)$ and $w \in \mathcal{L}(s)$. By definition of ${}^{P_1}||^{P_2}_\Gamma$ it must be that $v = \epsilon$ and $w = \epsilon$. By induction, this is equivalent to $n(r) \wedge n(s)$. □

The derivative of an expression r w.r.t. some symbol x, written $d_x(r)$ yields a new expression where the leading symbol x has been removed. In its definition, we write $(p) \Rightarrow r$ for: if p then r else ϕ.

Definition 9. *The derivative of $r \in R_\Sigma$ w.r.t. $x \in \Sigma$, written $d_x(r)$, is computed inductively as follows.*

$$
\begin{array}{lll}
d_x(\phi) & = \phi & \textit{(D1)} \\
d_x(\epsilon) & = \phi & \textit{(D2)} \\
d_y(x) & = (x = y) \Rightarrow \epsilon & \textit{(D3)} \\
d_x(r + s) & = d_x(r) + d_x(s) & \textit{(D4)} \\
d_x(r \cdot s) & = (d_x(r)) \cdot s + (n(r)) \Rightarrow d_x(s) & \textit{(D5)} \\
d_x(r^*) & = (d_x(r)) \cdot r^* & \textit{(D6)} \\
d_x(r^{P_1} ||_\Gamma^{P_2} s) & = (x \notin \Gamma) \Rightarrow (d_x(r)^{P_1} ||_\Gamma^{P_2} s) + (r^{P_1} ||_\Gamma^{P_2} d_x(s)) & \textit{(D7)} \\
 & \quad + (x \in \Gamma \wedge P_1 \cap P_2 = \emptyset) \Rightarrow (d_x(r)^{\emptyset} ||_\Gamma^{\emptyset} d_x(s)) & \textit{(D8)} \\
 & \quad + (x \in \Gamma \wedge P_1 \cap P_2 \neq \emptyset) \Rightarrow (d_x(r)^{P_1} ||_\Gamma^{P_2} d_x(s)) & \textit{(D9)} \\
 & \quad + (x \in \Gamma \wedge (P_1 \cup x) \cap P_2 = \emptyset) \Rightarrow (d_x(r)^{P_1 \cup x} ||_\Gamma^{P_2} s) & \textit{(D10)} \\
 & \quad + (x \in \Gamma \wedge P_1 \cap (P_2 \cup x) = \emptyset) \Rightarrow (r^{P_1} ||_\Gamma^{P_2 \cup x} d_x(s)) & \textit{(D11)}
\end{array}
$$

The definition extends to words and sets of words. We define $d_\epsilon(r) = r$ and $d_{xw}(r) = d_w(d_x(r))$. For $L \subseteq \Sigma^*$ we define $d_L(r) = \{d_w(r) \mid w \in L\}$.

We refer to the special case $d_{\Sigma^*}(r)$ as the set of descendants *of r. A descendant is either the expression itself, a derivative of the expression, or the derivative of a descendant.*

The first six cases (D1-6) correspond to Brzozowski's original definition [3]. As a minor difference we may concatenate with ϕ when building the derivative of a concatenated expression whose first component is not nullable. The new sub-cases (D7-11) for the general shuffle closely correspond to the sub-cases of Definition 5. For example, compare (D7) and (G3-4), (D8) and (G5), (D9) and (G6), (D10) and (G7), and lastly (D11) and (G8).

An easy induction shows that the derivative of a shuffle expression is again a shuffle expression.

Theorem 5 (Closure). *For any $r \in R_\Sigma$ and $x \in \Sigma$ we have that $d_x(r) \in R_\Sigma$.*

Brzozowski proved that the derivative of a regular expression denotes a left quotient. This result extends to shuffle expressions.

Theorem 6 (Left Quotients). *For any $r \in R_\Sigma$ and $x \in \Sigma$ we have that $\mathcal{L}(d_x(r)) = x \backslash \mathcal{L}(r)$.*

Proof. It suffices to consider the new case of general synchronous shuffling. We consider the direction $\mathcal{L}(d_x(r^{P_1} ||_\Gamma^{P_2} s)) \subseteq x \backslash \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s) x = \{w \mid x \cdot w \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)\}$.

Suppose $u \in \mathcal{L}(d_x(r^{P_1} ||_\Gamma^{P_2} s))$. We will verify that $x \cdot u \in \mathcal{L}(r^{P_1} ||_\Gamma^{P_2} s)$. We proceed by distinguishing among the following cases.

– Case $x \notin \Gamma$:
  By definition of the derivative operation, we find that either (D7a) $u \in \mathcal{L}(d_x(r)^{P_1}||_\Gamma^{P_2} s)$ or (D7b) $u \in \mathcal{L}(r^{P_1}||_\Gamma^{P_2} d_x(s))$.
  - Case (D7a):
    1. By definition $u \in v^{P_1}||_\Gamma^{P_2} w$ for some $v \in \mathcal{L}(d_x(r))$ and $w \in \mathcal{L}(s)$.
    2. By induction $x \cdot v \in \mathcal{L}(r)$.
    3. By observing the various cases for w (see (G2-3) in Definition 5) we follow that $x \cdot u \in x \cdot v^{P_1}||_\Gamma^{P_2} w$.
    4. Hence, $x \cdot u \in \mathcal{L}(r^{P_1}||_\Gamma^{P_2} s)$ and we are done.
  - Case (D7b): Similar to the above.

– Case $x \in \Gamma$:
  By definition of the derivative operation (D8) $u \in \mathcal{L}(d_x(r)^{\emptyset}||_\Gamma^{\emptyset} d_x(s))$ where $P_1 \cap P_2 = \emptyset$, or (D9) $u \in \mathcal{L}(d_x(r)^{P_1}||_\Gamma^{P_2} d_x(s))$ where $P_1 \cap P_2 \neq \emptyset$, or (D10) $u \in \mathcal{L}(d_x(r)^{P_1 \cup x}||_\Gamma^{P_2} s)$ where $(P_1 \cup x) \cap P_2 = \emptyset$, or (D11) $u \in \mathcal{L}(r^{P_1}||_\Gamma^{P_2 \cup x} d_x(s))$ where $P_1 \cap (P_2 \cup x) = \emptyset$.
  - Case (D8):
    1. By definition $u \in v^{\emptyset}||_\Gamma^{\emptyset} w$ for some $v \in \mathcal{L}(d_x(r))$ and $w \in \mathcal{L}(d_x(s))$.
    2. By induction $x \cdot v \in \mathcal{L}(r)$ and $x \cdot w \in \mathcal{L}(s)$.
    3. By case (G5) $x \cdot u \in x \cdot v^{P_1}||_\Gamma^{P_2} x \cdot w$.
    4. Hence, $x \cdot u \in \mathcal{L}(r^{P_1}||_\Gamma^{P_2} s)$ and we are done.
  - Case (D9): Similar to the above. Instead of (G5) we can apply (G6).
  - Case (D10):
    1. By definition $u \in v^{P_1 \cup x}||_\Gamma^{P_2} w$ for some $v \in \mathcal{L}(d_x(r))$ and $w \in \mathcal{L}(s)$.
    2. By induction $x \cdot v \in \mathcal{L}(r)$.
    3. By observing the various cases for w (see (G2) and (G7)) we follow that $x \cdot u \in x \cdot v^{P_1}||_\Gamma^{P_2} w$.
    4. Hence, $x \cdot u \in \mathcal{L}(r^{P_1}||_\Gamma^{P_2} s)$ and we are done.
  - Case (D11): Similar to the above.

The other direction $\mathcal{L}(d_x(r^{P_1}||_\Gamma^{P_2} s)) \supseteq \{x \cdot w \mid w \in \mathcal{L}(r^{P_1}||_\Gamma^{P_2} s)\}$ follows via similar reasoning. □

Based on the above result, we obtain a simple algorithm for membership testing. Given a word w and expression r, we exhaustively apply the derivative operation and on the final expression we apply the nullable test. That is $w \in \mathcal{L}(r)$ iff $n(d_w(r))$.

In general, it seems wasteful to repeatedly generate derivatives just for the sake of testing a specific word. A more efficient method is to construct a DFA via which we can then test many words. Brzozowski recognized that there is an elegant DFA construction method based on derivatives. Expressions are treated as states. For each expression and its derivative we find a transition.

For this construction to work we must establish that (1) the transitions implied by the derivatives cover all cases, and (2) the set of states remains finite. This is what we will consider next.

First, we establish (1) by verifying that each shuffle expression can be represented as a sum of its derivatives, extending another result of Brzozowski.

Theorem 7 (Representation). *For any* $r \in R_\Sigma$, $\mathcal{L}(r) = \mathcal{L}((n(r)) \Rightarrow \epsilon) \cup \bigcup_{x \in \Sigma} \mathcal{L}(x \cdot d_x(r))$.

Proof. Follows immediately from Lemma 1 and Theorem 6. □

States are descendants of expression r. Hence, we must verify that the set $d_{\Sigma^*}(r)$ is finite. In general, this may not be the case as shown by the following example

$$\begin{aligned} d_x(x^*) &= \epsilon \cdot x^* \\ d_x(\epsilon \cdot x^*) &= \phi \cdot x^* + \epsilon \cdot x^* \\ d_x(\phi \cdot x^* + \epsilon \cdot x^*) &= (\phi \cdot x^* + \epsilon \cdot x^*) + (\phi \cdot x^* + \epsilon \cdot x^*) \\ &... \end{aligned}$$

To guarantee finiteness, we need to consider expressions modulo *similarity*.

Definition 10 (Similarity). *We say that two expressions* $r, s \in R_\Sigma$ *are* similar*, written* $r \approx s$*, if one can be transformed into the other by applying the following identities:*

(I1) $r + s = s + r$ *(I2)* $r + (s + t) = (r + s) + t$ *(I3)* $r + r = r$

For $S \subseteq R_\Sigma$ *we write* $S/\approx$ *to denote the set of equivalence classes of all similar expressions in* S*.*

For the above example, we find that $(\phi \cdot x^* + \epsilon \cdot x^*) + (\phi \cdot x^* + \epsilon \cdot x^*) \approx \phi \cdot x^* + \epsilon \cdot x^*$ by application of (I3). To show an application of (I2), consider

$$\begin{aligned} d_x(x^* + x \cdot x^*) &= = \epsilon \cdot x^* + x^* \\ d_x(\epsilon \cdot x^* + x^*) &= (\phi \cdot x^* + \epsilon x^*) + \epsilon \cdot x^* \end{aligned}$$

Clearly, $(\phi \cdot x^* + \epsilon x^*) + \epsilon \cdot x^* \approx \phi \cdot x^* + \epsilon x^*$. Application of (I1) is omitted for brevity.

To verify (dis)similarity among descendants it suffices to apply identities (I1-3) at the top-level, i.e. highest position in the abstract syntax tree represenation of expressions. Top-level alternatives are kept in a list and sorted according to the number of occurrences of symbols. Any duplicates in the list are removed. Thus, the set $d_{\Sigma^*}(r)/\approx$ is obtained by generating dissimilar descendants, starting with $\{r\}$. That this generation step reaches a fix-point is guaranteed by the following result.

Theorem 8 (Finiteness). *For any* $r \in R_\Sigma$ *the set* $d_{\Sigma^*}(r)/\approx$ *is finite.*

Proof. It suffices to consider the new case of shuffle expressions. Our argumentation is similar to the case of concatenation in Brzozowski's original result. See proofs of Theorems 4.3(a) and 5.2 in [3]. It suffices to consider the new form $r^{P_1}||_\Gamma^{P_2} s$ which we will abbreviate by t.

By inspection of the definition of $d()$ on ${}^{P_1}||_\Gamma^{P_2}$ and application of identity (I2) (associativity) we find that all descendants of t can be represented as a sum of

expressions which are either of the shape ϕ or ${r'}^{P_1'}||_\Gamma^{P_2'} s'$ where r' is a descendant of r and s' is a descendant of s, but P_1' and P_2' are arbitrary subsets of Σ.

Thus, we can apply a similar argument as in Brzozowski's original proof to approximate the number of descendants of $r^{P_1}||_\Gamma^{P_2} s$ by the number of descendants of r and s.

Suppose $\sharp D_r$ denotes the number of all descendants of r and n is the number of elements in Σ. Then, we can approximate $\sharp D_t$ as follows:

$$\sharp D_t \leq 2^{\sharp D_r * \sharp D_s * 2^n * 2^n}$$

The exponent counts the number of different factors of the form ${r'}^{P_1}||_\Gamma^{P_2} s'$ and a sum corresponds to a subset of these factors. The factor $2^n * 2^n$ arises because P_1, P_2 range over subsets of Σ where $n = |\Sigma|$. The factor $\sharp D_s * \sharp D_r$ arises from the variation of r' and s' over the descendants of r and s, respectively.

As $\sharp D_r$ and $\sharp D_s$ are finite, by the inductive hypothesis, we obtain a very large, but finite bound on $\sharp D_t$. □

We summarize.

Definition 11 (Derivative-based DFA Construction). *For any $r \in R_\Sigma$ we define $\mathcal{D}(r) = (Q, \Sigma, q_0, \delta, F)$ where $Q = d_{\Sigma^*}(r)/\approx$, $q_0 = r$, for each $q \in Q$ and $x \in \Sigma$ we define $\delta(q, x) = d_x(q)$, and $F = \{q \in Q \mid n(q)\}$.*

Theorem 9. *For any $r \in R_\Sigma$ we have that $\mathcal{L}(r) = \mathcal{L}(\mathcal{D}(r))$.*

5.1 Discussion

Derivatives for free To obtain derivatives for the various shuffling variants in Section 3 we apply the following method. Each shuffling variant is transformed into its general synchronous shuffle representation as specified in Section 4. On the resulting expression we can then apply the derivative construction from Section 5.

For interleaving ($\|$), weakly ($| \sim |_\Gamma$) and strongly synchronized shuffling ($|||_\Gamma$) the transformation step is purely syntactic. For example, in case of weak synchronous shuffling expressions $r_1 | \sim |_\Gamma r_2$ are exhaustively transformed into ${r_1}^{\emptyset}||_\Gamma^{\emptyset} r_2$. The transformation step is more involved for synchronous composition ($|||$) as we must compute the alphabet of expressions (resp. the alphabet of the underlying languages). We define $\alpha(r) = \alpha(\mathcal{L}(r))$.

For plain regular expressions, we can easily compute the alphabet by observing the structure of expressions. For example, $\alpha(r^*) = \alpha(r)$. $\alpha(x) = \{x\}$. $\alpha(\phi) = \{\}$. $\alpha(\epsilon) = \{\}$. $\alpha(r + s) = \alpha(r) \cup \alpha(s)$. $\alpha(r \cdot s) = \alpha(r) \cup \alpha(s)$ if $\mathcal{L}(r), \mathcal{L}(s) \neq \{\}$. Otherwise, $\alpha(r \cdot s) = \{\}$. The test $\mathcal{L}(r) \neq \{\}$ can again be defined by observing the expression structure. We omit the details.

In the presence of shuffle expressions such as synchronous composition, it is not obvious how to appropriately extend the above structural definition. For example, consider $x \cdot x \cdot y|||x \cdot y$. We find that $\alpha(x \cdot x \cdot y) = \{x, y\}$ and $\alpha(x \cdot y)$ but $\alpha(x \cdot x \cdot y|||x \cdot y) = \{\}$ due to the fact that $x \cdot x \cdot y|||x \cdot y$ equals ϕ.

Hence, to compute the alphabet of some $r \in R_\Sigma$ we first convert r into a DFA M using the derivative-based automata construction. To compute the alphabet of M we use a variant of the standard emptiness check algorithm for DFAs. First, we compute all reachable paths from any of the final states to the initial state. To avoid infinite loops we are careful not to visit a transition twice on a path. Then, we obtain the alphabet of M by collecting the set of all symbols on all transitions along these paths.

Thus, the transformation of expressions composed of synchronous composition proceeds as follows. In the to be transformed expression, we pick any subexpression $r_1|||r_2$ where $r_1, r_2 \in R_\Sigma$. If there is none we are done. Otherwise, we must find r_1 and r_2 which have already been transformed, resp., do not contain any shuffling operator. Alphabets $\alpha(r_1)$ and $\alpha(r_2)$ are computed as described and subexpression $r_1|||r_2$ is replaced by $r_1 {}^{\Sigma}||^{\Sigma}_{\alpha(r_1)\cap\alpha(r_2)} r_2$. This process repeats until all synchronous composition operations have been replaced.

Specialization of derivative method For shuffling ($\|$) and strongly synchronized shuffling ($|||_\Gamma$), it is possible to derive specialized derivative operations. In the following, $R_\Sigma^{|\cdot|}$ denotes the subset of regular expressions restricted to shuffle expressions composed of $|\cdot|$ where $|\cdot|$ stands for any of the shuffling forms we have seen so far.

Theorem 10 (Derivatives Closure for Shuffling). *For any $r \in R_\Sigma^{\|}$ and $x \in \Sigma$ we have that $d_x(r) \in R_\Sigma^{\|}$.*

Proof. Recall that $\|$ can be expressed as ${}^{\emptyset}||^{\emptyset}_{\emptyset}$. By case analysis of Definition 9. Case (D7) applies only. □

Theorem 11 (Derivatives Closure for Strongly Synchronized Shuffling). *For any $r \in R_\Sigma^{|||_\Gamma}$ and $x \in \Sigma$ we have that $d_x(r) \in R_\Sigma^{|||_\Gamma}$.*

Proof. Recall that $|||_\Gamma$ can be expressed as ${}^{\Gamma}||^{\Gamma}_{\Gamma}$. Again by case analysis. This time only cases (D7) and (D9) apply. □

The above closure results show that the general derivative method in Definition 9 can be specialized for the case of asynchronous and strongly synchronized shuffling. The respective proofs describe the relevant cases.

For weak synchronous shuffling, we can no longer guarantee the closure property. For example, consider weak synchronous shuffling $| \sim |_\Gamma$ which is expressed by ${}^{\emptyset}||^{\emptyset}_{\Gamma}$. For expression $x \cdot z^{\emptyset}||^{\emptyset}_{\{x\}} y$. we find that

$$d_x(x \cdot z^{\emptyset}||^{\emptyset}_{\{x\}} y) = z^{\{x\}}||^{\emptyset}_{\{x\}} y + x \cdot z^{\emptyset}||^{\{x\}}_{\{x\}} \phi$$

The expression on the right-hand side is *not* part of $R_\Sigma^{|\sim|_\Gamma}$ due to subexpressions of the form $z^{\{x\}}||^{\emptyset}_{\{x\}} y$. However, these forms are necessary for correctness. Hence, to appropriately define derivatives for weak synchronous shuffling derivatives it is strictly necessary to enrich the expression language with general synchronous shuffling.

A similar observation applies to synchronous composition. For brevity, we omit the details.

6 Conclusion

Thanks to a general form of synchronous shuffling we can extend the notion of Brzozowski derivatives to various forms of shuffling which appear in the literature [2, 11, 4]. This enables the application of algorithms based on derivatives for shuffle expressions such as automata-based word recognition algorithms [10] and equality/containment checking [1, 6].

There are several avenues for future work. For example, it is well-known that associativity does not hold for synchronous composition. The work in [8] identifies sufficient conditions to guarantee associativity and other algebraic laws. It would be interesting to identify such conditions in terms of our general synchronous shuffling operation.

In another direction, it would be interesting to study in detail the impact of the various shuffling variants on the size of the derivative-automata. Earlier work [5] only considers the specific case of (asynchronous) shuffling.

References

1. Antimirov, V.M.: Rewriting regular inequalities. In: Proc. of FCT'95. LNCS, vol. 965, pp. 116–125. Springer-Verlag (1995)
2. ter Beek, M.H., Martín-Vide, C., Mitrana, V.: Synchronized shuffles. Theor. Comput. Sci. 341(1-3), 263–275 (2005)
3. Brzozowski, J.A.: Derivatives of regular expressions. J. ACM 11(4), 481–494 (1964)
4. Garg, V.K., Ragunath, M.T.: Concurrent regular expressions and their relationship to petri nets. Theor. Comput. Sci. 96(2), 285–304 (Apr 1992)
5. Gelade, W.: Succinctness of regular expressions with interleaving, intersection and counting. Theor. Comput. Sci. 411(31-33), 2987–2998 (Jun 2010)
6. Grabmayer, C.: Using proofs by coinduction to find "traditional" proofs. In: Proc. of CALCO'05. pp. 175–193. Springer-Verlag (2005)
7. Kumar, A., Verma, A.K.: A novel algorithm for the conversion of parallel regular expressions to non-deterministic finite automata. Applied Mathematics & Information Sciences 8, 95–105 (2014)
8. Latteux, M., Roos, Y.: Synchronized shuffle and regular languages. In: Jewels Are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa. pp. 35–44. Springer-Verlag, London, UK, UK (1999)
9. Lodaya, K., Mukund, M., Phawade, R.: Kleene theorems for product systems. In: Proc. of DCFS'11. LNCS, vol. 6808, pp. 235–247. Springer (2011)
10. Owens, S., Reppy, J., Turon, A.: Regular-expression derivatives reexamined. Journal of Functional Programming 19(2), 173–190 (2009)
11. de Simone, R.: Langages infinitaires et produit de mixage. Theor. Comput. Sci. 31, 83–100 (1984)
12. Stotts, P.D., Pugh, W.: Parallel finite automata for modeling concurrent software systems. J. Syst. Softw. 27(1), 27–43 (Oct 1994)